Roger Lopez
ansible-feedback@redhat.com
Abstract

This document provides the best practices to deploy Ansible Automation Platform 2.1.
Table of Contents

COMMENTS AND FEEDBACK ................................................................. 4

CHAPTER 1. OVERVIEW ................................................................. 5

CHAPTER 2. PREREQUISITES .......................................................... 8
  2.1. NODE REQUIREMENTS .................................................. 8
  2.2. NETWORK REQUIREMENTS ............................................ 11
  2.3. VALIDATION CHECKLIST FOR NODES ......................... 12

CHAPTER 3. ANSIBLE AUTOMATION PLATFORM CONTROLLER CONFIGURATION DETAILS ................................. 14
  3.1. NETWORK CONFIGURATION ........................................... 14
     3.1.1. Chrony Configuration ........................................... 14
  3.2. OS CONFIGURATION ................................................... 15
     3.2.1. Red Hat Subscription Manager ................................. 15
     3.2.2. User Accounts .................................................. 15
     3.2.3. Copying SSH keys to all nodes ............................... 16
     3.2.4. Configuring Firewall Settings ............................... 16

CHAPTER 4. ANSIBLE AUTOMATION PLATFORM CONTROLLER DATABASE CONFIGURATION DETAILS ............... 18
  4.1. CONFIGURING CONTROLLER DATABASE FIREWALL SETTINGS ........................................... 18

CHAPTER 5. ANSIBLE AUTOMATION PLATFORM EXECUTION AND HOP NODE CONFIGURATION DETAILS ..................... 19
  5.1. CONFIGURING FIREWALL SETTINGS FOR EXECUTION AND HOP NODES .................................. 19

CHAPTER 6. INSTALLING ANSIBLE AUTOMATION PLATFORM 2.1 ................................................................. 20

CHAPTER 7. PRIVATE AUTOMATION HUB ........................................ 24
  7.1. REQUIREMENTS FOR A HIGH AVAILABILITY AUTOMATION HUB ........................................... 24
     7.1.1. Enabling firewall services ....................................... 24
     7.1.2. Shared Filesystem ................................................ 24

CHAPTER 8. INSTALL PRIVATE AUTOMATION HUB ........................................ 26

CHAPTER 9. CENTRALIZED LOGGING .............................................. 30
  9.1. SETTING UP THE SPLUNK HTTP EVENT COLLECTOR (HEC) .................................................. 30
  9.2. CONFIGURING ANSIBLE AUTOMATION PLATFORM AUTOMATION CONTROLLER ......................... 34
  9.3. VERIFYING EVENTS SENT TO SPLUNK ................................ 35

CHAPTER 10. CONFIGURATION CONSISTENCY ACROSS MULTI ANSIBLE AUTOMATION PLATFORM DEPLOYMENTS ..................................................... 38
  10.1. WHY CONFIGURATION AS CODE? ................................. 38
  10.2. WHAT IS CONFIGURATION AS CODE? ............................ 38
  10.3. WHAT ARE GIT WEBHOOKS? ........................................ 38
  10.4. WORKING WITH WEBHOOKS ....................................... 39
     10.4.1. GitHub webhook setup ......................................... 39

APPENDIX A. ABOUT THE AUTHOR .................................................... 48

APPENDIX B. CONTRIBUTORS ............................................................ 49

APPENDIX C. SUPPLEMENTAL .......................................................... 50
  C.1. CONFIGURING AUTOMATION CONTROLLER PLAYBOOK ....... 50
  C.2. GROUP_VARS/ALL.YML VARS FILE FOR AUTOMATION CONTROLLER CONFIGURATION (USER CREATION SAMPLE) 50
COMMENTS AND FEEDBACK

In the spirit of open source, we invite anyone to provide feedback and comments on any reference architecture. Although we review our papers internally, sometimes issues or typographical errors are encountered. Feedback allows us to not only improve the quality of the papers we produce, but allows the reader to provide their thoughts on potential improvements and topic expansion to the papers. Feedback on the papers can be provided by emailing ansible-feedback@redhat.com. Please refer to the title within the email.
CHAPTER 1. OVERVIEW

The Ansible Automation Platform 2.1 reference architecture provides an opinionated setup of deploying a highly available Ansible Automation Platform environment. It provides a step-by-step deployment procedure with the latest best practices to install and configure Ansible Automation Platform 2.1. It is best suited for system and platform administrators looking to deploy Ansible Automation Platform.

A pictorial representation of the environment used in this reference environment is shown in Figure 1.1, “Reference Architecture Overview”
The image above consists of two sites, *Ansible Site 1* and *Ansible Site 2* for high availability. Site one is an active environment while site two is a passive environment. Each site consists of:

- three node automation controller cluster with one PostgreSQL database
- three node automation hub cluster with one PostgreSQL database
- Two execution nodes per automation controller cluster
Access to console.redhat.com services such as Red Hat Insights and Service Catalog

In order to achieve HA for the PostgreSQL databases, configuration as code in conjunction with Git webhooks are used when push or merge events are triggered on a Git repository which in turn will configure the specified event on both Ansible Site 1 and Ansible Site 2.

Finally, to ensure logging consistency, a highly available centralized logging environment is installed that both Ansible Automation Platform environments will use.
CHAPTER 2. PREREQUISITES

The installation of this highly available Ansible Automation Platform 2.1 reference environment uses the following on each site:

- Three control plane nodes
- One control plane database node
- Two execution nodes
- Three automation hub nodes
- One automation hub database node
- Shared filesystem for automation hub installation (/var/lib/pulp)

**NOTE**
These nodes are not required to be physical servers.

2.1. NODE REQUIREMENTS

**Table 2.1. execution nodes**

<table>
<thead>
<tr>
<th>Execution nodes</th>
<th>Required</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM</td>
<td>16Gb</td>
<td></td>
</tr>
<tr>
<td>CPUs</td>
<td>4</td>
<td>Runs automation. Increase memory and CPU to increase capacity for running more forks</td>
</tr>
</tbody>
</table>

**Table 2.2. automation controller nodes**

<table>
<thead>
<tr>
<th>Control nodes</th>
<th>Required</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM</td>
<td>16Gb</td>
<td></td>
</tr>
<tr>
<td>CPUs</td>
<td>4</td>
<td>Processes events and runs cluster jobs including project updates and cleanup jobs. Increasing CPU and memory can help with job event processing.</td>
</tr>
</tbody>
</table>
Disk: service node

40Gb dedicated hard disk space

- **controller**: dedicate a minimum of 20 GB to `/var/` for file and working directory storage
- Storage volume should be rated for a minimum baseline of 1500 IOPS.
- Projects are stored on control and hybrid, and for the duration of jobs, also on execution nodes. If the cluster has many large projects, consider having twice the GB in `/var/lib/awx/projects`, to avoid disk space errors.

Disk: database node

20Gb dedicated hard disk space

- 150 GB+ recommended
- Storage volume should be rated for a high baseline IOPS (1500 or more).

Browser

A currently supported version of Mozilla FireFox or Google Chrome

Database

PostgreSQL version 12

### Table 2.3. automation hub nodes

<table>
<thead>
<tr>
<th>automation hub nodes</th>
<th>Required</th>
<th>Notes</th>
</tr>
</thead>
</table>
| RAM                   | 8Gb minimum | - 8 GB RAM (minimum and recommended for Vagrant trial installations)
- 8 GB RAM (minimum for external standalone PostgreSQL databases)
- For capacity based on forks in your configuration, see additional resources |
## CPUs
2 minimum
- For capacity based on forks in your configuration, see additional resources

## Disk: service node
40Gb dedicated hard disk space
- Storage volume should be rated for a minimum baseline of 1500 IOPS.

## Disk: database node
20Gb dedicated hard disk space
- 150 GB+ recommended
- Storage volume should be rated for a high baseline IOPS (1500 or more).

## Browser
A currently supported version of Mozilla FireFox or Google Chrome

## Database
PostgreSQL version 12

### Table 2.4. database nodes

<table>
<thead>
<tr>
<th>database nodes</th>
<th>Required</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM</td>
<td>16GB</td>
<td></td>
</tr>
<tr>
<td>CPUs</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Disk</td>
<td>20Gb dedicated hard disk space</td>
<td>Minimum dedicated disk space 20Gb</td>
</tr>
<tr>
<td></td>
<td></td>
<td>150 GB+ recommended</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Storage volume should be rated for a high baseline IOPS (1500 or more)</td>
</tr>
</tbody>
</table>
NOTE

All automation controller data is stored in the PostgreSQL database. Database storage increases with the number of hosts managed, number of jobs run, number of facts stored in the fact cache, and number of tasks in any individual job. For example, a playbook run every hour (24 times a day) across 250 hosts, with 20 tasks will store over 800,000 events in the database every week.

If not enough space is reserved in the database, old job runs and facts will need cleaned on a regular basis. Refer to Management Jobs in the Automation Controller Administration Guide for more information

2.2. NETWORK REQUIREMENTS

Ansible Automation Platform 2.1 requires at least one network across all nodes within the Ansible Automation Platform cluster.

In order to access the Ansible Automation Platform dashboard, a browser that can access the network that the control plane nodes reside on is required. If you wish to access the Ansible Automation Platform dashboard externally, ensure to add a public IP address to your control plane nodes.

It is recommended that network administrators setup a static DHCP address for all nodes and an appropriate DNS record. This ensures each node’s IP address remains constant in the absence of a DHCP server. To use static IP addresses, reserve the IP addresses with an infinite lease.

For the purposes of this reference architecture, setup of a DHCP server, setting up DNS records and setting up a load balancer is out of scope.

A network administrator should reserve at least the following number of IP addresses, including:

1. One IP address for each control plane node.
2. One IP address for each execution node.
3. One IP address for each automation hub node.
4. One IP address for the control plane database.
5. One IP address for the automation hub database.
6. One IP address for the load balancer automation controller cluster address for Ansible Automation Platform Site 1.
7. One IP address for the load balancer automation controller cluster address for Ansible Automation Platform Site 2.
8. One IP address for the load balancer private automation hub cluster address for Ansible Automation Platform Site 1.
9. One IP address for the load balancer private automation hub cluster address for Ansible Automation Platform Site 2.

This reference environment reserves 12 IP addresses per site.

The following table provides an example of Ansible Site 1 of the reference environment.
<table>
<thead>
<tr>
<th>Usage</th>
<th>Host Name</th>
<th>IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control Plane 1</td>
<td>controlplane-1.site1.example.com</td>
<td>192.168.0.10</td>
</tr>
<tr>
<td>Control Plane 2</td>
<td>controlplane-2.site1.example.com</td>
<td>192.168.0.11</td>
</tr>
<tr>
<td>Control Plane 3</td>
<td>controlplane-3.site1.example.com</td>
<td>192.168.0.12</td>
</tr>
<tr>
<td>Control Plane Database</td>
<td>controlplane-db.site1.example.com</td>
<td>192.168.0.13</td>
</tr>
<tr>
<td>Control Plane Cluster address</td>
<td>controlplane-cluster.site1.example.com</td>
<td>192.168.0.14</td>
</tr>
<tr>
<td>Execution Node 1</td>
<td>executionnode-1.site1.example.com</td>
<td>192.168.0.15</td>
</tr>
<tr>
<td>Execution Node 2</td>
<td>executionnode-2.site1.example.com</td>
<td>192.168.0.16</td>
</tr>
<tr>
<td>Automation Hub Node 1</td>
<td>automationhub-1.site1.example.com</td>
<td>192.168.0.17</td>
</tr>
<tr>
<td>Automation Hub Node 2</td>
<td>automationhub-2.site1.example.com</td>
<td>192.168.0.18</td>
</tr>
<tr>
<td>Automation Hub Node 3</td>
<td>automationhub-3.site1.example.com</td>
<td>192.168.0.19</td>
</tr>
<tr>
<td>Automation Hub Database</td>
<td>automationhub-db.site1.example.com</td>
<td>192.168.0.20</td>
</tr>
<tr>
<td>Automation Hub Cluster</td>
<td>automationhub-cluster.site1.example.com</td>
<td>192.168.0.21</td>
</tr>
</tbody>
</table>

### 2.3. VALIDATION CHECKLIST FOR NODES

The following is a summary of all the requirements:

- 16 GB of RAM for controller nodes and execution nodes
- 8 GB of RAM for private automation hub nodes
- 4 CPUs for controller nodes and execution nodes
- 2 CPUs for private automation hub nodes
- 20 GB+ disk space for database node
- 40 GB+ disk space for non database nodes
DHCP reservations use infinite leases to deploy the cluster with static IP addresses.

DNS records for all nodes

Red Hat Enterprise Linux 8.4 or later 64-bit (x86) installed for all nodes

Chrony configurationed for all nodes

ansible-core version 2.11 or later installed for all nodes
CHAPTER 3. ANSIBLE AUTOMATION PLATFORM 
CONTROLLER CONFIGURATION DETAILS

This reference architecture focuses on the deployment of Ansible Automation Platform 2.1 using automation mesh on Red Hat Enterprise Linux 8.4 x86_64. The configuration is intended to provide a comprehensive Ansible Automation Platform solution. The key solution components covered within this reference architecture consists of:

- Red Hat Enterprise Linux 8.4
- Ansible Automation Platform 2.1
- automation mesh
- private automation hub

3.1. NETWORK CONFIGURATION

3.1.1. Chrony Configuration

Each Ansible Automation Platform node in the cluster must have access to an NTP server. The `chronyd` is a daemon for synchronization of the system clock. It can synchronize the clock with NTP servers. This ensures that when cluster nodes use SSL certificates that require validation, they don’t fail if the date and time between the nodes are not in sync.

On all the nodes,

1. If not installed, install `chrony` as follows

   ```bash
   # dnf install chrony --assumeyes
   ```

2. Edit the `/etc/chrony.conf` file with a text editor such as `vi`

   ```bash
   # vi /etc/chrony.conf
   ```

3. Locate the following public server pool section, and modify to include the appropriate servers. Only one server is required, but three is recommended. The `iburst` option is added to speed up the time that it takes to properly sync with the servers.

   ```bash
   # Use public servers from the pool.ntp.org project.
   # Please consider joining the pool (http://www.pool.ntp.org/join.html).
   server <ntp-server-address> iburst
   ```

4. Save all the changes within the `/etc/chrony.conf` file.

5. Start and enable that the `chronyd` daemon is started when the host is booted.

   ```bash
   # systemctl --now enable chronyd.service
   ```

6. Verify the `chronyd` daemon status.

   ```bash
   # systemctl status chronyd.service
   ```
3.2. OS CONFIGURATION

3.2.1. Red Hat Subscription Manager

The `subscription-manager` command registers a system to the Red Hat Network (RHN) and manages the subscription entitlements for a system. The `--help` option specifies on the command line to query the command for the available options. If the `--help` option is issued along with a command directive, then options available for the specific command directive are listed.

To use Red Hat Subscription Management for providing packages to a system, the system must first register with the service. In order to register a system, use the `subscription-manager` command and pass the `register` command directive. If the `--username` and `--password` options are specified, then the command does not prompt for the RHN Network authentication credentials.

An example of registering a system using `subscription-manager` is shown below.

```
# subscription-manager register --username [User] --password '[Password]'
The system has been registered with id: abcd1234-ab12-ab12-ab12-481ba8187f60
```

After a system is registered, it must be attached to an entitlement pool. For the purposes of this reference environment, the Red Hat Ansible Automation Platform is the pool chosen. Identify and subscribe to the Red Hat Ansible Automation Platform entitlement pool, the following command directives are required.

```
# subscription-manager list --available | grep -A8 "Red Hat Ansible Automation Platform"
---
Subscription Name:   Red Hat Ansible Automation Platform, Premium (5000 Managed Nodes)
Provides:            Red Hat Ansible Engine
                                Red Hat Single Sign-On
                                Red Hat Ansible Automation Platform
SKU:                 MCT3695
Contract:            <contract>
Pool ID:             <pool_id>
Provides Management: No
Available:           9990
Suggested:           1
Service Type:        L1-L3
Roles:

# subscription-manager attach --pool <pool_id>
Successfully attached a subscription for: Red Hat Ansible Automation Platform, Premium (5000 Managed Nodes)

# subscription-manager repos --enable=ansible-automation-platform-2.1-for-rhel-8-x86_64-rpms
```

3.2.2. User Accounts

Prior to the installation of Ansible Automation Platform 2.1, it is recommended to create a non-root user with `sudo` privileges for the deployment process. This user is used for:

- SSH connectivity
- passwordless authentication during installation
For the purposes of this reference environment, the user **ansible** was chosen, however, any user name would suffice.

On all nodes, create a user named **ansible** and generate an **ssh** key.

1. Create a non-root user
   ```bash
   # useradd ansible
   ```

2. Set a password for your **ansible** user.
   ```bash
   # passwd ansible
   ```

3. Generate an **ssh** key as the **ansible** user.
   ```bash
   $ ssh-keygen -t rsa
   ```

4. Disable password requirements when using **sudo** as the **ansible** user
   ```bash
   # echo "ansible ALL=(ALL) NOPASSWD:ALL" | sudo tee -a /etc/sudoers.d/ansible
   ```

3.2.3. Copying SSH keys to all nodes

With the **ansible** user created, as the **ansible** user, copy the **ssh** key to all the nodes. This ensures that when the Ansible Automation Platform installation runs, it can **ssh** to all the nodes without a password.

This can be done using the **ssh-copy-id** command as follows:

```bash
$ ssh-copy-id ansible@hostname.example.com
```

**NOTE**

If running within a cloud provider, you may need to instead create an `~/.ssh/authorized_keys` file containing the public key for the **ansible** user on all your nodes and set the permissions to the `authorized_keys` file to only the owner (**ansible**) having read and write access (permissions 644).

3.2.4. Configuring Firewall Settings

Firewall access and restrictions play a critical role in securing Ansible Automation Platform 2.1 environment. The use of Red Hat Enterprise Linux 8.4 defaults to using **firewalld**, a dynamic firewall daemon. **firewalld** works by assigning network zones to assign a level of trust to a network and its associated connections and interfaces.

It is recommended that firewall settings be configured to permit access to the appropriate services and ports for a success Ansible Automation Platform 2.1 installation.

On all nodes, ensure that **firewalld** is installed, started and enabled.

1. Install the **firewalld** package
   ```bash
   # dnf install firewalld --assumeyes
   ```
2. Start the `firewalld` service
   
   ```
   # systemctl start firewalld
   ```

3. Enable the `firewalld` service
   
   ```
   # systemctl enable firewalld
   ```
CHAPTER 4. ANSIBLE AUTOMATION PLATFORM
CONTROLLER DATABASE CONFIGURATION DETAILS

4.1. CONFIGURING CONTROLLER DATABASE FIREWALL SETTINGS

For a successful Ansible Automation Platform installation, one of the prerequisites is to enable the database port on the database node. The port used is configured by the `pg_port` within your `inventory` file.

Snippet of inventory file

```plaintext
pg_port='5432'
```

Within your database node, as the `ansible` user, set the `firewalld` port to be used for installation.

1. Ensure that `firewalld` is running.

   ```bash
   $ sudo systemctl status firewalld
   ```

2. Add the `firewalld` port on your controller database node (e.g. port 5432)

   ```bash
   $ sudo firewall-cmd --permanent --zone=public --add-port=5432/tcp
   ```

3. Reload `firewalld`

   ```bash
   $ sudo firewall-cmd --reload
   ```

4. Confirm that the port is open

   ```bash
   $ sudo firewall-cmd --list-ports
   ```
5.1. CONFIGURING FIREWALL SETTINGS FOR EXECUTION AND HOP NODES

For a successful Ansible Automation Platform installation, one of the prerequisites is to enable the automation mesh port on the mesh nodes (execution and hop nodes). The default port used for the mesh networks on all the nodes is set to 27199/tcp, however, it can be configured to use a different port by specifying `receptor_listener_port` as each node’s variable within your inventory file.

Snippet of inventory file

```
receptor_listener_port=27199
```

NOTE

For this reference environment all the Ansible Automation Platform 2 controller nodes are designated as node type **control**. If control nodes are designated as **hybrid** nodes (default node type), they require mesh port (default: 27199/tcp) to be enabled.

Within your hop and execution node(s), as the **ansible** user:

1. Ensure that **firewalld** is running.
   ```
   $ sudo systemctl status firewalld
   ```

2. Add the **firewalld** port on your hop and execution nodes (e.g. port 27199)
   ```
   $ sudo firewall-cmd --permanent --zone=public --add-port=27199/tcp
   ```

3. Reload **firewalld**
   ```
   $ sudo firewall-cmd --reload
   ```

4. Confirm that the port is open
   ```
   $ sudo firewall-cmd --list-ports
   ```
CHAPTER 6. INSTALLING ANSIBLE AUTOMATION PLATFORM 2.1

The installation of Ansible Automation Platform 2.1 takes advantage of automation controller and automation mesh to provide a simple, secure and flexible method of handling your automation workloads.

Automation controller provides the control plane for automation through its UI, Restful API, RBAC workflows and CI/CD integrations.

Automation mesh is an overlay network that provides the ability to ease the distribution of work across a large and dispersed collection of workers through nodes that establish peer-to-peer connections with each other using existing networks.

With automation mesh in place, you can:

- Dynamically scale cluster capacity without downtime
- Decouple the execution and control plane
- Re-route execution to a different path when outages may exist automatically

The following steps provide a step-by-step on deploying a clustered Ansible Automation Platform 2.1 using automation mesh.

NOTE

The following installation is done on Ansible Site 1 of this reference environment Ansible Automation Platform cluster. After completion, ensure to follow the same steps for Ansible Site 2.

On controlplane-1.site1.example.com, as the ansible user, perform the following:

1. Download Ansible Automation Platform 2.1 Setup tar
   ```bash
   ansible-automation-platform-setup-2.1.0-1.tar.gz
   ```

2. Untar the ansible-automation-platform-setup-2.1.0-1.tar.gz
   ```bash
   $ tar zxvf ansible-automation-platform-setup-2.1.0-1.tar.gz
   ```

3. Change directory into ansible-automation-platform-setup-2.1.0-1.tar.gz
   ```bash
   cd ansible-automation-platform-setup-2.1.0-1/
   ```

4. Backup the existing inventory file
   ```bash
   $ cp inventory inventory.bkup
   ```

5. Install the ansible-core package
   ```bash
   $ sudo dnf install ansible-core --assumeyes
   ```
6. Modify the inventory to include with the appropriate information about your environment. Below is an example of this reference environment.

```yaml
[automationcontroller]
controlplane-1.site1.example.com ansible_connection=local
controlplane-2.site1.example.com
controlplane-3.site1.example.com

[automationcontroller:vars]
node_type=control
peers=execution_nodes

[execution_nodes]
executionnode-1.site1.example.com peers=executionnode-2.site1.example.com
executionnode-2.site1.example.com

[database]
controldatabase.site1.example.com

[all:vars]
#Handled by Ansible Vault
admin_password=""

pg_host='controldatabase.site1.example.com'
pg_port='5432'
pg_database='awx'
pg_username='awx'

#Handled by Ansible Vault
pg_password=""
pg_sslmode='prefer'

registry_url='registry.redhat.io'
registry_username='myusername'

#Handled by Ansible Vault
registry_password=""
```

1. control nodes run project and inventory updates and system jobs, but not execution jobs. Execution capabilities are disabled on these nodes.
2. Peer relationships define node-to-node connections. Setting peer relationship between control plane nodes and execution nodes.
3. Setting peer relationships between the execution nodes.
4. Set the node that will install the PostgreSQL database for the controller installation.
5. Set the password for admin user to access the UI upon install completion.
6. Set the PostgreSQL host (database node).
7. Set the PostgreSQL port to use for the database node.
Set the password for the PostgreSQL database.

Execution Environment images are downloaded and included in your installation. Proper credentials required to download the images.

User credential for access to registry_url.

Password credential for access to registry_url.

7. Create a file labeled `credentials.yml` that will store the encrypted credentials.

```
$ cat credentials.yml

admin_password: my_long_admin_pw
pg_password: my_long_pg_pw
registry_password: my_long_registry_pw
```

8. Encrypt the `credentials.yml` file using `ansible-vault`.

```
$ ansible-vault encrypt credentials.yml

New Vault password:
Confirm New Vault password:
Encryption successful
```

9. Verify the `credentials.yml` file is encrypted

```
$ cat credentials.yml

$ANSIBLE_VAULT;1.1;AES256
3638363965356238653431633333961383336306465366666306531666310a3730623013337633938313033
3431353438396261363230376163666326393265306234383961363965365643365616233313365
3636616639313864300a3532393734333133396134653263393130356335653534635653865386
5363163346343835346432766386235361366632613634332313163343639396461326561643336
3406335340393564626463034383966336232303365383763
```
10. Run the `setup.sh` for installation of Ansible Automation Platform 2.1 and pass the `credentials.yml`, as well as, the `--ask-vault-pass` option

```
$ ANSIBLE_BECOME_METHOD='sudo' ANSIBLE_BECOME=True
ANSIBLE_HOST_KEY_CHECKING=False ./setup.sh -e @credentials.yml -- --ask-vault-pass
```

**NOTE**

The following `ANSIBLE*_` variables are set to ensure a successful install.

**NOTE**

For more information regarding the different values that can be set within your inventory file, visit: Setting up the inventory file

11. Login to the Ansible Automation Platform dashboard, e.g. controlplane-cluster.site1.example.com

12. Activate your Ansible Automation Platform subscription via Subscription manifest or username/password.

**NOTE**

The steps to generate your subscription manifest can be found Appendix D, Generate a subscription manifest
CHAPTER 7. PRIVATE AUTOMATION HUB

Automation Hub is the central place for the certified collections. It functions as the main source of trusted, tested and supported content. Private automation hub provides automation developers the ability to collaborate and publish their own automation content and streamline delivery of Ansible code within their organization.

Private automation hub in Ansible Automation Platform 2.1 primarily delivers support for automation execution environments. Execution environments are a standardized way to define, build and distribute the environments that the automation runs in. In a nutshell, automation execution environments are container images that allow for easier administration of Ansible by the platform administrator.

7.1. REQUIREMENTS FOR A HIGH AVAILABILITY AUTOMATION HUB

7.1.1. Enabling firewall services

As part of a highly available automation hub environment, due to the requirement of using a shared filesystem, the following firewall services should be enabled to ensure successful mounting of the filesystem as shown in the next section.

On each automation hub node, as the ansible user:

1. Ensure the following firewalld services (`nfs, mountd, rpc-bind`) are enabled.
   
   ```sh
   $ sudo firewall-cmd --zone=public --add-service=nfs
   $ sudo firewall-cmd --zone=public --add-service=mountd
   $ sudo firewall-cmd --zone=public --add-service=rpc-bind
   ```

2. Reload firewalld for changes to take effect.

   ```sh
   $ sudo firewall-cmd --reload
   ```

3. Verify the firewalld services are enabled

   ```sh
   $ sudo firewall-cmd --get-services
   ```

7.1.2. Shared Filesystem

A high availability automation hub requires you to have a shared file system already set up for your environment.

Before you run the Ansible Automation Platform installer, verify that the `/var/lib/pulp` directory exists across your cluster as a result of your shared file system installation. The Ansible Automation Platform installer will return an error if `/var/lib/pulp` is not detected in one of your nodes, causing your highly available automation hub setup to fail.

On each automation hub node, as the ansible user:

1. Create the `/var/lib/pulp` directory

   ```sh
   $ sudo mkdir /var/lib/pulp
   ```

2. Mount the shared filesystem (this reference environment uses an NFS share)
$ sudo mount -t nfs4 <nfs_share_ip_address>:/var/lib/pulp

3. Confirm the shared filesystem is successfully mounted via:

$ df -h
CHAPTER 8. INSTALL PRIVATE AUTOMATION HUB

With all the prerequisites set and in place, the final step is to modify the inventory file for a clustered automation hub installation and run setup.sh.

Within this reference environment, the controlplane-1.site1.example.com system is used to run the setup.sh for the clustered automation hub environment as it can access all the nodes.

NOTE

The following installation is done on site Ansible Site 1 of this reference environment Ansible Automation Platform cluster. After completion, ensure to follow the same steps for Ansible Site 2.

On controlplane-1.site1.example.com, as the ansible user, perform the following:

1. Change directory into ansible-automation-platform-setup-2.1.0-1.tar.gz

   cd ansible-automation-platform-setup-2.1.0-1/

2. Backup the existing inventory file

   $ cp inventory inventory-controller.bkup

3. Modify the inventory file to include with the appropriate information about your environment. Below is an example of this reference environment.

   [automationcontroller]

   [automationcontroller:vars]

   [execution_nodes]

   [automationhub]
   ahub-1.site1.example.com 1
   ahub-2.site1.example.com
   ahub-3.site1.example.com

   [database]
   ahub-db.site1.example.com 2

   [servicescatalog_workers]

   [sso]

   [all:vars]
   admin_password="
   pg_host="
   pg_port="
   pg_database="
   pg_username="
pg_password=""  
pg_sslmode=""  # set to 'verify-full' for client-side enforced SSL

registry_url = 'registry.redhat.io'  
registry_username = 'myusername'

#Handled by Ansible vault  
registry_password=""

receptor_listener_port = 27199

#Handled by Ansible vault  
automationhub_admin_password=""

automationhub_pg_host = 'ahub-db.site1.example.com'  
automationhub_pg_port = '5432'

automationhub_pg_database = 'automationhub'  
automationhub_pg_username = 'automationhub'

#Handled by Ansible vault  
automationhub_pg_password=""

automationhub_pg_sslmode = 'prefer'

ssos_console_admin_password=""

 automation hub nodes.

 Set the node that will install the PostgreSQL database for the automation hub installation.

 Execution Environment images are downloaded and included in your installation. Proper credentials required to download the images.

 User credential for access to registry_url.

 Password credential for access to registry_url.

 Set the password for admin user to access the UI upon install completion.

 Set the PostgreSQL host (database node).

 Set the PostgreSQL port to use for the automation hub database node.

 Set the password for the PostgreSQL database.

---

**NOTE**

All the information regarding the Ansible Automation Platform controller installation was removed.

---

4. Create a file labeled *credentials_ah.yml* that will store the encrypted credentials.
5. Encrypt the `credentials_ah.yml` file using `ansible-vault`.

   ```bash
   $ ansible-vault encrypt credentials_ah.yml
   
   New Vault password:
   Confirm New Vault password:
   Encryption successful
   
   WARNING
   Store your encrypted vault password in a safe place.
   ```

6. Verify the `credentials_ah.yml` file is encrypted

   ```bash
   $ cat credentials_ah.yml
   
   $ANSIBLE_VAULT;1.1;AES256
   3638363965356238653431633333396138333363064653364656138313534353135303764646
   16539
   37653930633030653234666333064623236306531666310a3730623031333763396338313
   03033
   3431353438962613632303761636632623932653062343839613639653636433656162333
   13365
   3636616639313864300a353239373433313339613465326339313035635653534643565386
   53631
   6334643438353464323766386235363163666261363433323131633436393964613265616
   43336
   34306335343039356462643303438396633623230365383763
   
   7. Run the `setup.sh` for installation of Ansible Automation Platform 2.1 and pass the `credentials_ah.yml`, as well as, the `--ask-vault-pass` option

   ```bash
   $ ANSIBLE_BECOME_METHOD='sudo' ANSIBLE_BECOME=True ANSIBLE_HOST_KEY_CHECKING=False ./setup.sh -e @credentials_ah.yml -- --ask-vault-pass
   
   NOTE
   The following `ANSIBLE*`_ variables are set to ensure a successful install.
NOTE

For more information regarding the different values that can be set within your inventory file, visit: Setting up the inventory file

8. Login to the private automation hub dashboard, e.g. automationhub-cluster.site1.example.com
CHAPTER 9. CENTRALIZED LOGGING

When we think about logging, the first thought that comes to mind for most is its ability to troubleshoot a particular issue. As technology continues to evolve and the enormous amounts of data that many applications capture, logs play a vital role in not only capturing data but to allow for operational intelligence methods to be applied.

Ansible Automation Platform provides a logging feature that allows for the capability to send detailed logs to several kinds of third party external log aggregation services. Services connected to this data feed serve as a useful means in gaining insight into automation controller usage or technical trends. The data can be used to analyze events in the infrastructure, monitor for anomalies, and correlate events from one service with events in another.

The types of data that are most useful to automation controller are job fact data, job events/job runs, activity stream data, and log messages. The data is sent in JSON format over a HTTP connection using minimal service-specific tweaks engineered in a custom handler or via an imported library.

The logging feature within Ansible Automation Platform is currently set up to easily work with Splunk, Logstash, Loggly, Sumologic and provides an other option if you’d like to use a different third party external log aggregation service.

For the purposes of this reference environment, the focus is on using Splunk Enterprise 8.2.2 to setup centralized logging across both our Ansible Automation Platform sites.

NOTE

Installation of Splunk Enterprise is out of scope for this reference architecture. For details, visit: How to install Splunk Enterprise

9.1. SETTING UP THE SPLUNK HTTP EVENT COLLECTOR (HEC)

The HTTP Event Collector is an endpoint that lets developers send application events directly to the Splunk platform via HTTP or HTTPS using a token-based authentication model.

In order to use HEC, the first step is to enable HEC within our Splunk deployment.

As a Splunk admin,

1. Log into your Splunk dashboard.
2. Click on Settings → Data Inputs → HTTP Event Collector.
3. Click on Global Settings in the upper right corner.
4. Select Enabled (default) in the All Tokens toggle options.
5. Choose a Default Source Type for all HEC tokens, e.g. _json
6. Checkmark the Enable SSL if your Splunk deployment uses HTTPS.
7. Set a dedicated port for HTTP input via the HTTP Port Number. Defaults to 8088.
8. Click Save.
WARNING

Ensure that port 8088 (or the assigned port) is open on your Splunk deployment.

With HEC enabled in our Splunk deployment, generate an HEC token that is going to be used for automation controller authentication with Splunk.

While there are different methods on how to configure your token for receiving data over HTTP, the following are the configuration settings used for this reference environment.

NOTE

For more information on HEC visit: HTTP Event Collector

1. Click Settings → Add Data.
2. Select Monitor at the bottom of the page.
3. Click HTTP Event Collector.
4. Enter your token name in the Name field, e.g. AAP
5. Change Source Type to Select and enter _json in the dropdown.
6. Within Index, create an Index labeled ansible.
7. Add the newly created Index into the selected items box.

Input Settings
Optionally set additional input parameters for this data input as follows:

Source type
The source type is one of the default fields that the Splunk platform assigns to all incoming data. It tells the Splunk platform what kind of data you've got, so that the Splunk platform can format the data intelligently during indexing. And it's a way to categorize your data, so that you can search it easily.

Index
The Splunk platform stores incoming data as events in the selected index. Consider using a "sandbox" index as a destination if you have problems determining a source type for your data. A sandbox index lets you troubleshoot your configuration without impacting production indexes. You can always change this setting later. Learn More.

Select Allowed Indexes
Add all
Selected item(s) remove all

Available item(s)
- ansible
- history
- main
- summary

Default Index
- ansible

Create a new index

8. Click Review at the top right corner.
Review

Input Type ....................... Token
Name ............................ AAP
Source name override ....... N/A
Description ..................... N/A
Enable indexer acknowledg No
Output Group .................... N/A
Allowed indexes .............. ansible

Default index ..................... ansible
Source Type ..................... _json
App Context ...................... search

9. Click Submit.

10. Store the created Token value as it will be used to authenticate Splunk on Ansible Automation Platform

Token has been created successfully.
Configure your inputs by going to Settings > Data Inputs
11. Click the **Start Search** green button and copy the supplied sample query for later use.

```
source=\"http:AAP\" (index=\"ansible\") sourcetype=\"_json\"
```

### 9.2. CONFIGURING ANSIBLE AUTOMATION PLATFORM AUTOMATION CONTROLLER

With HEC enabled and an HEC token created, our Splunk environment is ready to receive events from Ansible Automation Platform.

The final step is to configure our Ansible Automation Platform automation controller cluster to use the Splunk environment for centralized logging as shown below.

On each Ansible Automation Platform environment,

1. Login as an **admin** user to the Ansible Automation Platform dashboard.
2. Scroll to the bottom of the page and click on **Settings**.
3. Under **System**, select **Logging settings**.

![Ansible Automation Platform dashboard with Settings highlighted](image)

4. Within **Logging Aggregator** enter the location that the logs should be sent to.
   a. This Splunk environment uses `https://splunk.example.com:8088/services/collector/event`

   **NOTE**

   Modify your location Protocol (HTTP/HTTPS) and the port if not using the default 8088.

5. Within **Logging Aggregator Type** select **splunk** from the drop down.
6. Within **Logging Aggregator Password/Token**, copy and paste the HEC token previously created.
7. Within **Logging Aggregator Protocol**, select **HTTPS/HTTP** from the drop down.
8. Within **Logging Aggregator Level Threshold**, select the appropriate level of logging for your environment, e.g. *INFO*

**Figure 9.2. Reference Environment Logging Setup**

![Logging Aggregator Setup](image)

**NOTE**

The settings above include the minimum to achieve logging with Splunk. Adjust the logging settings to best fit your environment.

Repeat the logging feature configuration on site two once complete. This ensures both sites are using the same centralized logging environment.

**9.3. VERIFYING EVENTS SENT TO SPLUNK**

Lastly, verify that the Ansible Automation Platform events are being appropriately sent to Splunk. In order to accomplish this, an ad-hoc command via Ansible Automation Platform automation controller is run to confirm.

Within the Ansible Automation Platform dashboard,

1. Under **Resources → Inventories**, select **Demo Inventory**.

2. Within **Details**, select **Hosts**.

3. Click on the **Run Command** button.

4. Within the **Run command Details** window section, select the module **ping** from the drop down and click **Next**.

5. Within the **Run command Execution Environment** section, select **Default execution environment** and click **Next**.
6. Within the **Run command Credential** section, select **Demo Credential** and click **Next**.

7. Within the **Run command Preview** section, click the blue **Launch** button.

The output provided should look as follows:

```
localhost | SUCCESS => {
  "changed": false,
  "ping": "pong"
}
```

Now head over to your Splunk dashboard and within the Search see if the Ansible Automation Platform event triggered within Splunk.

```
splunk>enterprise Apps ▼
Search Analytics Datasets Reports Alerts Dashboards

New Search

source="http:My Token" (index="ansible") sourcetype="_json" SUCCESS

1 of 1 event matched No Event Sampling ▼
```

The search should show one event similar to the following
Repeat the same verification process for each Ansible Automation Platform site setup.
CHAPTER 10. CONFIGURATION CONSISTENCY ACROSS MULTI ANSIBLE AUTOMATION PLATFORM DEPLOYMENTS

10.1. WHY CONFIGURATION AS CODE?

Traditionally, when we think about having consistency across multiple Ansible Automation Platform deployment sites, we think of using database replication to copy the data from one Ansible Automation Platform environment and importing that data to another Ansible Automation Platform site.

This could be done with scripts, specialized software, manual export of the database followed by an import the Ansible Automation Platform database to the new site, or even running custom Ansible playbooks that do something similar.

While there are many different methods, typically, these methods have cons associated with them.

For example, database replication requires investments in storage infrastructure and concurrency could be hard to achieve without the right expertise.

10.2. WHAT IS CONFIGURATION AS CODE?

Configuration as code is defined as the practice of managing configuration files in a repository. In the case of Ansible Automation Platform, these configuration files establish the settings we wish to apply across our Ansible Automation Platform environments.

By storing and managing Ansible Automation Platform configuration files as code, we can:

- standardize the settings being applied to all our Ansible Automation Platform environments
- inherit the benefits of version control of our configurations
- easily scale additional Ansible Automation Platform deployments to use the same configuration settings
- easily track changes of the configuration settings making fixing issues easier

While configuration as code provides numerous benefits and best practices that can be applied across an Ansible Automation Platform environment, in order to streamline and automate the delivery of the configurations to our multiple Ansible Automation Platform sites automatically, immediately and consistently, we must pair our configuration as code solution with Git webhooks.

10.3. WHAT ARE GIT WEBHOOKS?

Git webhooks are defined as a method for notifications to be delivered to an external web server whenever certain actions occur on a repository or organization.

For example, when a repository is updated, this could trigger an event that could trigger CI builds, deploy an environment, or in our case, update configurations to our Ansible Automation Platform environments.

With the solution of configuration as code and Git webhooks, we can setup a Ansible Automation Platform work flow that immediately updates all of our Ansible Automation Platform sites simultaneously with the exact configurations across all the platforms.

In effect, removing the overhead of having to maintain database backups or enable an expensive database replication solution while achieving the pros of those solutions.
With this methodology in mind, the following sections focus on how an admin can take advantage of configuration as code to ensure consistency across multiple Ansible Automation Platform environments using Git webhooks.

10.4. WORKING WITH WEBHOOKS

A webhook provides the ability to execute specified commands between apps over the web. Ansible automation controller currently provides webhook integration with GitHub and GitLab. This reference environment describes the procedure for setting up a webhook in automation controller using GitHub.

10.4.1. GitHub webhook setup

Automation controller has the ability to run jobs based on a triggered webhook event. The following provides the step-by-step for that setup.

Create a GitHub personal access token

1. Generate a personal access token (PAT) for use with automation controller.
   a. In the profile settings of your GitHub account, click **Settings**.
   b. Below the Personal settings, click **Developer Settings**.
c. In the Developer settings, click **Personal access tokens**.

d. From the Personal access tokens screen, click **Generate new token** button.

e. When prompted, enter your GitHub account password to continue.

f. In the **Note** field, enter a brief description about what this PAT will be used for.

g. In the **Expiration** drop down, select **No expiration**.
In the Scope fields, automation controller webhook only needs repo scope access, with the exception of invites. For information about other scopes, click the link right above the table to access the docs.

Note

AAP_Site1 Webhook

What's this token for?

Expiration *

No expiration ✗ The token will never expire!

GitHub strongly recommends that you set an expiration date for your token to help keep your information secure. Learn more

Select scopes
Scopes define the access for personal tokens. Read more about OAuth scopes.

<table>
<thead>
<tr>
<th>Scope</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>repo</td>
<td>Full control of private repositories</td>
</tr>
<tr>
<td>repo_status</td>
<td>Access commit status</td>
</tr>
<tr>
<td>repo_deployment</td>
<td>Access deployment status</td>
</tr>
<tr>
<td>public_repo</td>
<td>Access public repositories</td>
</tr>
<tr>
<td>repo:invite</td>
<td>Access repository invitations</td>
</tr>
<tr>
<td>security_events</td>
<td>Read and write security events</td>
</tr>
</tbody>
</table>

Click the Generate Token button at the bottom of the page.

WARNING

Once the token is generated, make sure you copy the PAT, as it will be used by Ansible Automation Platform automation controller cluster in a later step. You will not be able to access this token again in GitHub.

Create a GitHub repository

Once we have our PAT in place, the next step is to create a Git repository that will be triggered by our GitHub webhooks when changes are made to the repository.

Since the focus is to be able to easily update or modify our automation controller configurations, this reference environment takes advantage of an Ansible collection named redhat_cop.controller_configuration.

This Ansible collection allows for easy interaction with Ansible controller servers via Ansible roles using the controller collection modules.

With the help of this collection, we can use these pre-existing roles to modify or update our automation controller environments across all our sites.
NOTE

The redhat_cop.controller_configuration collection is a community project and officially not supported by Red Hat.

1. In GitHub, create a new repository.
   a. Ensure to select the appropriate Owner and provide a Repository name.
      i. This reference environment uses the repository name aap_refarch.
   b. Provide an optional description to describe your repository.
   c. Set your repository to public or private.
      i. This reference environment sets the repository to Public.
   d. Initialize the repository with an Add a README file

2. Click on the create repository button.

Within the newly created GitHub repository, three files are created.

- playbook.yml - A playbook with all the appropriate roles to update or modify our automation controller.
- requirements.yml - A requirements.yml file with all the collections required to run the playbook.
- group_vars/all.yml - An all.yml file containing all the variables we wish to modify for a specific configuration (role).

The purpose of these three files is to provide a method that when the group_vars/all.yml is modified and updated to the repository, this will kick off a job template within our Ansible Automation Platform automation controller cluster to make the appropriate change or update.

NOTE

A sample of these three files can be found within the Appendix C, Supplemental section of this guide or https://github.com/ansible/aap_refarch

Setting up resource credentials within Ansible Automation Platform automation controller cluster

Prior to using our repository as configuration as code, setup the appropriate credential resources within our Ansible Automation Platform sites.

This ensures that as new projects, work flows and job templates are created, we can easily attach those credentials to those resources.

The following steps show how to create our two required credentials.

One credential associated with the GitHub (PAT) token we created previously.

The other credential associated with the Ansible Automation Platform site that will get updated. Each Ansible Automation Platform site will get updated by its own environment, e.g. site1 gets updated by sitel.

On each Ansible Automation Platform site:
1. Login to your Ansible Automation Platform automation controller dashboard

2. Under Resources→Credentials click the blue Add button.
   
   a. Provide a Name, e.g. GitHub PAT
   
   b. Select GitHub Personal Access Token as the Credential Type.
   
   c. Within Type Details, add the secret using the previously generated token from GitHub.

3. Click Save.

   **NOTE**
   
   Make note of the name of this credential, as it will be used in the job template that posts back to GitHub.

With the GitHub PAT credential set, create a second credential for the the existing Ansible Automation Platform environment.

1. Under Resources→Credentials click the blue Add button.
   
   a. Provide a Name, e.g. AAP_Site1
   
   b. Provide Red Hat Ansible Automation Platform as the Credential Type.
   
   c. Within Type Details, add the appropriate details
      
      i. Red Hat Ansible Automation Platform, e.g. controlplane-cluster.site1.example.com
      
      ii. Username, e.g. admin
      
      iii. Password, e.g. redhat
   
   d. If the SSL certificate is signed, ensure to select Verify SSL within Options.

2. Click Save.

Repeat this section on Ansible Automation Platform site two.

   **NOTE**
   
   controlplane-cluster.site1.example.com is used as this reference environment takes advantage of a load balancer. Setup of a load balancer is out of scope for this reference environment.

**Creating the configuration as code project**

The purpose of the configuration as code project is to create a work flow that contains a job template that automatically runs every time an update to the configuration as code repository takes place.

This ensures that as you make changes such as setting variables, the Git repository playbook runs the appropriate roles for the numerous automation controller configurations.

This section creates a project, work flow and job template to achieve the above.
Within your Ansible Automation Platform site one dashboard,

1. Under **Resources→ Projects** click the blue **Add** button.
2. Provide a **Name**, e.g. Configuration as Code Project
3. Select **Default** as the Organization
4. Select **Default execution environment** as the **Execution Environment**
5. Select **Git** as the **Source Control Credential Type**
6. **Within Type Details,**
   a. Add the **Source Control URL** (your GitHub repository)
7. **Within Options**
   a. Select **Clean, Delete, Update Revision on Launch**
8. Click **Save.**

Next, create a work flow template.

1. Under **Resources→ Templates** click the blue **Add → Add workflow template.**
2. Provide a **Name**, e.g. Configuration as Code Workflow
3. **Within Options,** checkmark **Enable Webhook**
   a. **Within Webhook details,** select **GitHub** as the **Webhook Service.**
   b. **Within Webhook details,** select your GitHub PAT token previously created as the **Webhook Credential,** e.g. GitHub PAT
4. Click **Save.**
5. **Within the Please click the Start button to begin window,** click the **Save** at the top right corner.
6. Copy the **Webhook URL** and the **Webhook Key** as they will be used later.

Repeat the above process on Ansible Automation Platform site two.

**Enabling GitHub Webhooks for the Repository**

With our Ansible Automation Platform Workflow template created and our GitHub repository with the required files in place, the next step is to enable webhooks for our repository, e.g. *aap_refarch*.

1. At the homepage of your GitHub repository, select the **Settings** tab.
2. **Within the Settings tab,** select **Webhooks.**
3. Within the **Webhooks** section, select the **Add webhook** button.

4. Enter the **Payload URL** (Webhook URL of the workflow).

5. Change the **Content type** drop down to `application/json`.

6. Enter the **Secret** (Webhook key of the workflow).

7. Leave the defaults to use push events, and click the button **Add webhook**.

**WARNING**

By default, GitHub verifies SSL certificates when delivering payloads. If your automation controller SSL certificates are not signed, ensure to disable SSL verification.

Repeat the above process on Ansible Automation Platform site two.

**Creating the configuration as code job template**

The job template to be created for our configuration as code project runs the `playbook.yml` file automatically every time an update to the Git repository takes places.

This ensures as configuration changes are desired, we can use the Ansible Automation Platform controller API to make those changes appropriately. When we combine the same methodology across all our Ansible Automation Platform sites, all the changes become global across the configured sites.

1. Under **Resources→Templates** click the blue **Add→Add job template**.

2. Provide a **Name**, e.g. Configuration as Code Job

3. Select **Run** as the **Job Type**

4. Select **Demo Inventory** as the **Inventory**

5. Select **Configuration as Code Project** as the **Project**

6. Select **Default execution environment** as the **Execution Environment**

7. Select `playbook.yml` as the **Playbook**
8. Select Credentials and switch the category from Machine to Red Hat Ansible Automation Platform.

9. Select the appropriate credential for the Ansible Automation Platform site, e.g. AAP_Site1.

10. Within Options, select Enable webhook.

11. Select GitHub as the Webhook Service.

12. Select your GitHub PAT token previously created as the Webhook Credential, e.g. GitHub PAT.

13. Click Save.

Repeat the above process on Ansible Automation Platform site two.

**Updating the created Configuration as Code Workflow**

Previously, the Configuration as Code workflow was created. The purpose of this workflow is to ensure that the Configuration as Code Project is always in sync and that the Configuration as Code Job runs the configuration as code playbook whenever changes are made to the repository.

1. Under Resources → Templates, select your template. e.g Configuration as Code Workflow.

2. Within the Details section, select the Visualizer tab and click the green Start.

3. For Node Type select Project Sync and select the appropriate project, e.g. Configuration as Code Project and click Save.

4. Hover over the Configuration as Code Project and select the plus "+" symbol.

5. Within the Add Node window, select Always as to when this node should be executed and click Next.

6. Select the Configuration as Code Job as the Node Type and click Save.

7. Once brought back to the Visualizer, select the Save button at the top right corner.

**Verifying Configuration as Code setup**

With many of the roles already included in the redhat_cop.controller_configuration, a simple method to verify if everything is working as expected is to create a user by updating the `group_vars/all.yml` file with the appropriate yaml.

If we wanted to create a user, we would take a closer look at the users role documentation to further see all the variables at our disposable.

After further review, we notice creating a non-admin user, named John would look as follows:

**group_vars/all.yml**

```yaml
---
controller_user_accounts:
  - user: "john"
    is_superuser: false
    password: "redhat"
```
Once the `group_vars/all.yml` file is updated with the above and pushed to the Git repository, a job within Ansible Automation Platform automation controller should start.

This can be verified by selecting **Jobs** under the **Views** section within the automation controller dashboard. The job in reference should have the number of the job followed by the job name. For this reference environment, you’d see something similar to **8-Configuration as Code Job** with the **Type Playbook Run**.

![Screenshot of Automation Controller Jobs](image)

Once the job is complete, it will have a status of **Successful**.

Logging out as admin and relogging in as the user **john** with the password **redhat** should confirm that the job was successful.

If you followed this entire chapter and set the webhooks appropriately for each Ansible Automation Platform site, you will notice that the user john was created simultaneously by the corresponding Ansible Automation Platform platform.

While this example is a simplified one, many of the configurations (roles) that allow for the configuration of Ansible Automation Platform automation controller are provided within the **redhat_cop.controller_configuration**.

Additional examples may be found at: **Example configs**.

While the Ansible collection has many of the key roles you’ll ever need, you are not limited to only what can be found within this Ansible collection. New roles can be implemented or created by your organization to further enhance your capabilities.
APPENDIX A. ABOUT THE AUTHOR

Roger Lopez
Roger Lopez is a Principal Technical Marketing Manager bringing 10+ years of computer industry experience delivering high-value solutions used by our sales, marketing and engineering teams to develop best practice documentation & methods for internal and external customers. He is a Red Hat Certified Engineer (RHCE) with experience building solutions around Ansible, OpenShift and OpenStack.
# APPENDIX B. CONTRIBUTORS

We would like to thank the following individuals for their time and patience as we collaborated on this process. This document would not have been possible without their many contributions.

<table>
<thead>
<tr>
<th>Contributor</th>
<th>Title</th>
<th>Contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Landon Holley</td>
<td>Senior Consulting Architect</td>
<td>Technical Review</td>
</tr>
<tr>
<td>Ajay Chenampara</td>
<td>Principal Specialist Solution Architect</td>
<td>Technical Review</td>
</tr>
<tr>
<td>Will Tome</td>
<td>Principal Specialist Solution Architect</td>
<td>Technical Review</td>
</tr>
<tr>
<td>Sean Sullivan</td>
<td>Senior Consultant</td>
<td>Technical Review</td>
</tr>
<tr>
<td>Vinny Valdez</td>
<td>Senior Managing Architect</td>
<td>Technical Review</td>
</tr>
<tr>
<td>Chris Budzilowicz</td>
<td>Principal Technical Writer</td>
<td>Content Review</td>
</tr>
<tr>
<td>Phil Avery</td>
<td>Senior Ansible Specialist Solution Architect</td>
<td>Technical Review</td>
</tr>
</tbody>
</table>
APPENDIX C. SUPPLEMENTAL

C.1. CONFIGURING AUTOMATION CONTROLLER PLAYBOOK

---

- name: Playbook to configure ansible Controller post installation
  hosts: localhost
  connection: local
  vars:
    controller_validate_certs: false
  collections:
    - awx.awx
    - redhat_cop.controller_configuration

  roles:
    - {role: settings, when: controller_settings is defined, tags: settings}
    - {role: organizations, when: controller_organizations is defined, tags: organizations}
    - {role: labels, when: controller_labels is defined, tags: labels}
    - {role: users, when: controller_user_accounts is defined, tags: users}
    - {role: teams, when: controller_teams is defined, tags: teams}
    - {role: credential_types, when: controller_credential_types is defined, tags: credential_types}
    - {role: credentials, when: controller_credentials is defined, tags: credentials}
    - {role: credential_input_sources, when: controller_credential_input_sources is defined, tags: credential_input_sources}
    - {role: notification_templates, when: controller_notifications is defined, tags: notification_templates}
    - {role: projects, when: controller_projects is defined, tags: projects}
    - {role: execution_environments, when: controller_execution_environments is defined, tags: execution_environments}
    - {role: applications, when: controller_applications is defined, tags: applications}
    - {role: inventories, when: controller_inventories is defined, tags: inventories}
    - {role: instance_groups, when: controller_instance_groups is defined, tags: instance_groups}
    - {role: project_update, when: controller_projects is defined, tags: projects}
    - {role: inventory_sources, when: controller_inventory_sources is defined, tags: inventory_sources}
    - {role: inventory_source_update, when: controller_inventory_sources is defined, tags: inventory_sources}
    - {role: hosts, when: controller_hosts is defined, tags: hosts}
    - {role: groups, when: controller_groups is defined, tags: inventories}
    - {role: job_templates, when: controller_job_templates is defined, tags: job_templates}
    - {role: workflow_job_templates, when: controller_workflows is defined, tags: workflow_job_templates}
    - {role: schedules, when: controller_schedules is defined, tags: schedules}
    - {role: roles, when: controller_roles is defined, tags: roles}

C.2. GROUP_VARS/ALL.YML VARS FILE FOR AUTOMATION
CONTROLLER CONFIGURATION (USER CREATION SAMPLE)

---

controller_user_accounts:
  - user: "colin"
    is_superuser: false
    password: "redhat"
C.3. REQUIREMENTS.YML FOR AUTOMATION CONTROLLER CONFIGURATION

```yaml
collections:
- name: redhat_cop.controller_configuration
- name: awx.awx
```
APPENDIX D. GENERATE A SUBSCRIPTION MANIFEST

In order to successfully generate your subscription manifest:

1. Login to access.redhat.com and click Subscriptions at the top left corner.

2. Select the Subscriptions Allocations tab.

3. Select the button labeled New Subscription Allocation

4. Within the Create a New subscription allocation page, enter a Name and for Type select Subscription Asset Manager 1.4 from the dropdown box and click Create.

5. Once created, select the Subscriptions tab and select the Ansible Automation Platform subscription and the number of entitlements you wish to provide.

6. Click on the Details tab, review the number of entitlements are correct and select the Export Manifest button.
7. Return to the Ansible Automation Platform dashboard, and import `manifest.zip` and click Next.

8. (Recommended) Enable user analytics and Insights for Ansible Automation Platform by providing your Red Hat or Red Hat Satellite credentials and click Next.

9. Accept the end user license agreement.
APPENDIX E. REFERENCES

- Red Hat Ansible Automation Platform Installation Guide
- Aggregating Ansible Tower logs to Splunk by Joseph Tejal
- Centralize your Automation Logs with Ansible Tower and Splunk Enterprise by Leonardo Araujo
- Tower Logging and Aggregation
- Set up and use HTTP Event Collector in Splunk Web
- Splunk - Managing Indexers and Clusters of Indexers
- Fun with Private Automation Hub by Daniel Leroux
- Fun with Private Automation Hub Part II by Daniel Leroux
- Automation Controller Workflow Deployment as Code by Sean Sullivan
# APPENDIX F. REVISION HISTORY

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Author(s)</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2-0</td>
<td>2022-05-03</td>
<td>Anshul Behl,Roger Lopez</td>
<td>- Added hop and execution node automation mesh firewalld port requirements</td>
</tr>
<tr>
<td>1.1-0</td>
<td>2022-04-11</td>
<td>Roger Lopez</td>
<td>- Fixed error in overview image where arrow pointing to wrong location.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Added database node minimum requirements to Ch.2 Prerequisites</td>
</tr>
<tr>
<td>1.0-0</td>
<td>2021-12-02</td>
<td>Roger Lopez</td>
<td>- Initial Release</td>
</tr>
</tbody>
</table>